DEVELOPMENT OF AN AUTOMATIC DIFFERENTIATION VERSION OF THE FPX ROTOR CODE

Final Technical Report
of
Contract NAS 1-19935, Task No. 11

by

Hong Hu

Hampton University
Hampton, Virginia

August 1996

Prepared for

National Aeronautics and Space Administration

DEVELOPMENT OF AN AUTOMATIC DIFFERENTIATION VERSION OF THE FPX ROTOR CODE

Final Technical Report
of
Contract NAS 1-19935, Task No. 11

by

Hong Hu

Hampton University
Hampton, Virginia

August 1996

Prepared for

National Aeronautics and Space Administration

PREFACE

The work presented in this report is performed under contract NAS 1-19935, Task No. 11 from National Aeronautics and Space Administration, Langley Research Center. Dr. Henry E. Jones is the Technical Monitor, who has provided helpful suggestions and guidance during this investigation.

SUMMARY

The ADIFOR2.0 automatic differentiator is applied to the FPX rotor code along with the grid generator GRGN3. The FPX is an eXtended Full-Potential CFD code for rotor calculations. The automatic differentiation version of the code is obtained, which provides both non-geometry and geometry sensitivity derivatives. The sensitivity derivatives via automatic differentiation are presented and compared with divided difference generated derivatives. The study shows that automatic differentiation method gives accurate derivative values in an efficient manner.

CONTENTS

	Page
PREFACE	i
SUMMARY	ii
1. INTRODUCTION	1
2. METHODOLOGY OF THE CFD CODE - FPX	1
3. AUTOMATIC DIFFERENTIATION METHOD	2
4. DEVELOPMENT OF AD-VERSION OF FPX CODE	3
5. COMPUTATIONAL RESULTS	5
5.1 Accurancy	5
5.2 Memory Requirement and Computational Efficiency	6
6. CONCLUDING REMARKS	6
REFERENCES	14

1. INTRODUCTION

The recent advances in Computational Fluid Dynamics (CFD) have provided accurate and detailed flow solutions. These advances have generated interest in integrating CFD code into design methods. Success of integration of CFD into design depends on an efficient and accurate CFD code and an efficient and accurate method for calculating Sensitivity Derivatives (SD), coupled with an efficient design algorithm.

Sensitivity derivatives are defined as the derivatives of system response with respect to independent design variables. In other words, changes in the system outputs are related to the changes in the system inputs through a sensitivity derivative matrix. In the past, the SD matrix has been computed by divided difference (DD), direct differentiation, or symbolic differentiation methods. The accurancy of the DD method is hard to assess where the optimum step size is unknown. The direct differentiation and symbolic differentiation methods require considerable amount of work in comparison with the development of the original code, particularly when the system includes an advanced CFD model. The results of computing SD matrix to advanced CFD codes using Automation Differentiation (AD) method have been reported recently for non-gemotry [1,2] and geometry derivatives also[3-6]. These results demonstrated the feasibility of obtaining the exact SD via AD method.

In this report, the application of automatic differentiation method to an advanced CFD rotor code, FPX, is presented. The obtained AD code provides both non-geometry and geometry sensitivity derivatives. In the following sections, the CFD code and the automatic differentiation source translator ADIFOR2.0 are driefly discussed, which are followed by applications to the CFD code and the study of the SD results.

2. METHODOLOGY OF THE CFD CODE - FPX

While in the fixed-wing aerodynamic computational community more and more expensive and complex Euler and Navier-Stokes methods are used recently, potential methods still serve as a major analysis tool in the rotary-wing aerodynamic computational community. The eXtended Full-Potential (FPX) rotor code[7] is one such accurate and efficient potential method, which represents an industry standard for rotary-wing computations. The FPX code is a modified and enhanced version of Full-Potential Rotor (FPR) code[8]. The FPX code solves three-dimensional unsteady full-potential equation in a strong conservative form using an implicit approximate-factorization finite-difference scheme with entropy and viscosity corrections. The code (either FPR or FPX) has been used in various helicopter hover and forward flight cases, including Blade-Vortex Interaction (BVI) calculations[9], nonlinear acoustic analysis[10] and coupling with the comprehensive helicopter code CAMRAD[11]. The application of the code produces excellent results. The code is also highly optimized and a typical steady run takes 2-15 minutes on a Cray-YMP supercomputer.

The FPX/FPR codes solve the unsteady three-dimensional full-potential equation in a strong conservation form using an implicit finite-difference scheme for flow around rotor blade. It is less expensive than either Euler or Navier- Stokes method, and yet produces accurate solutions for various helicopter flows without significant separations.

The unsteady, three-dimensional full-potential equation in strong conservation form in blade-fixed body-conforming coordinates (ξ, η, ζ, τ) is written as

$$\frac{\partial}{\partial \tau}(\frac{\rho}{J}) + \frac{\partial}{\partial \xi}(\frac{\rho U}{J}) + \frac{\partial}{\partial \eta}(\frac{\rho V}{J}) + \frac{\partial}{\partial \zeta}(\frac{\rho W}{J}) = 0 \tag{1}$$

with

$$\rho = \{1 + \frac{\gamma - 1}{2}[-2\Phi_\tau - (U + \xi_t)\Phi_\xi - (V + \eta_t)\Phi_\eta - (W + \zeta_t)\Phi_\zeta]\}^{\frac{1}{\gamma - 1}} \tag{2}$$

where Φ is the velocity potential, U, V and W are contravariant velocity components, ρ is the density, and J is the grid Jacobian.

The FPX/FPR codes solve Eq. (1) using an implicit finite-difference scheme, where the time-derivative is repleced by a first-order backward differencing and the spatial-derivatives are repleced by second-order central differencing. The resulting differenc equation is approximately factored into three operators L_ξ, L_η, and L_ζ in ξ, η and ζ directions, respectively,

$$L_\xi L_\eta L_\zeta(\Phi^{n+1} - \Phi^n) = RHS \tag{3}$$

The detail of the scheme is presented in References 7 and 8.

The FPX is the substantially modified version of the FPR code. Both entropy and viscosity corrections are included in the FPX code. The entropy correction potential formulation accounts for the shock produced entropy to enhance physical modeling capabilities for strong shock cases. Either a two-dimensional or a three-dimensional boundary layer model is coupled with the FPX code to acoount for viscosity effects. In addition, an axial flow capability is added into the FPX code to treat tilt-rotors in forward flight. In addition to the O-H grid topology, an H-H grid topology is added as well. More recently, the Vorticity Embedding (VE) is incorporated into the FPX code to enhance the prediction capability of parallel blade-vortex interactions[12].

A grid generation package GRGN3 is used to generate C-H mesh around rotor blade. The blade surface is defined by an input file. The far-field boundary of the mesh is set at a fixed number of chords from blade surface. The mesh points are generated between blade surface and far-filed boundary.

3. AUTOMATIC DIFFERENTIATION METHOD

The automatic differentiation technique generates a set of derivatives of outputs with respect to inputs of the souce code. This is achieved by line-by-line differentiation, moving

from one line to the next and linking the derivatives through chain rule as required by the variable dependencies from the beginning to the end of the source code. In contrast to the DD approximation method, AD does not incur truncation error and hence the AD produces exact derivative value for non-iterative method at least. The resulting AD results are usually obtained with the working accurancy of the original function evaluation.

The AD technique is implemented into some forms of automatic differentiators. Automatic Differentiation In FORtran Version 2.0 (ADIFOR2.0) of Bischof et. al[13] is one of such differentiators, which uses source-transformation approach to provide the derivatives. The source-transformation approach is based on the fact that each statement in a Fortran source code is executed on a computer as an elementary operation. ADIFOR2.0 applies the chain rule

$$\frac{\partial f(s(t))}{\partial t} = \frac{\partial f(s)}{\partial s}\frac{\partial s}{\partial t} \tag{4}$$

over and over again to the composition of those elementary operations, such as addition and multiplications, to calculate derivative information of f exactly and in a completely mechanical manner. In this way, ADIFOR2.0 translates the Fortran 77 source code into an auxiliary code which computes both f and its derivative.

ADIFOR2.0 employs a hydrid approach of the forward and reverse modes of automatic differentiation. In this hybrid approach, for each statement ADIFOR2.0 accumulates the partial derivatives of the left-hand side variable with respect to the right-hand side variables, and then apply the forward mode to propagate the total derivatives according to the chain rule. This approach results in substantial decerase in complexity of the generated code compared with fully forward mode. Moreover, ADIFOR2.0 generated code provides the directional derivative computation possibilities. Instead of producing SD matrix J_{SD}, ADIFOR2.0 produces $J_{SD} \cdot S$, where the "seed matrix" S is initialized by the user.

ADIFOR2.0 is based on a source translator paradigm and designed for large-scale codes as well. ADIFOR2.0 has several advantages over other existing automatic differentiators. First, ADIFOR2.0 is very general and supports almost all statements of Fortran 77 code; it supports functions with branches and loops as well. Second, ADIFOR2.0 produces a plain Fortran 77 derivative code and hence it is computer-device independent. Third, ADIFOR2.0 is efficient in that it preserves the source code development effort.

4. DEVELOPMENT OF AD-VERSION OF FPX CODE

In the present work, ADIFOR2.0 is applied to FPX flow solver along with the grid generator GRGN3, which results in a sensitivity derivative version of the code for both non-geometry and geometry derivatives. The flow solver along with grid generator is considered as a "black-box" with input X and output F. Both X and F may be scalar or vector. The input X for the "black-box" may consists of various types of variables, such as free-stream flow conditions, airfoil/wing geometry, material properties of the fluids,

algorithm parameters, mesh size, and so on. The output F may also consists of various types of variables, such as aerodynamic performance, solution accurancy and so on. In the present work, the output F is chosen as lifting coefficient C_L, drag coefficient C_D and moment coefficient C_M; the input X is angle of attack α, blade-tip Mach number M_{tip} and blade-section geometry.

If only non-geometry derivatives are needed, application of ADIFOR2.0 to flow solver alone is enough. The ADIFOR2.0 can process entire FPX solution algorithm and produce a new version of the code (SD code) with same input and output as the original flow solver plus required derivatives, such as $\partial(C_L, C_D, C_M)/\partial(\alpha, M_{tip})$.

However if geometry derivatives are also needed, both the flow solver and the grid generator must be passed to ADIFOR2.0 as input. There are two ways to produce geometry derivatives. One way is to process flow solver and grid generator individually, and then link the derivatives together to get geometry derivatives[3-5]. For example if ($\partial C_L/\partial$airfoil section) is the desired SD, one must obtain ($\partial C_L/\partial$grid) from flow solver and (∂grid/∂airfoil section) from the grid generator; and then use the chain rule to compute ($\partial C_L/\partial$airfoil section) as follow:

$$\frac{\partial C_L}{\partial \text{airfoil section}} = \frac{\partial C_L}{\partial \text{grid}} \cdot \frac{\partial \text{grid}}{\partial \text{airfoil section}} \tag{5}$$

This is a natural way to compute geometry derivatives, since the grid generator is usually separeted from the flow solver as an independent code.

The other way is to combine grid generator with flow solver, and then input the combined grid generator and flow solver to ADIFOR2.0 to generate geometry derivatives in one step as it was done in Ref. 6. The present work employs this method. The nature of this method is identical with the former method, except that in this method Eq. (5) is carried out automatically within the SD code via ADIFOR2.0 and hence is more straightforward than the former method.

The grid generator GRGN3 is combined into the flow solver FPX, where the additional storage is created to provide link between the grid generator and the flow solver instead of using I/O statement as required by ADIFOR2.0 source translator. Once a single grid generator - flow solver code with an appropriate date link is obtained and some minor modifications to the code are made, application of ADIFOR2.0 becomes a simple and straightforward manner. A new version of the code for both non-geometry and geometry derivatives is obtained. The resulting SD code is then assembled into a working code by linking a driver (the main program initializing "seed matrix"and calling the SD code) and the ADIntrinsics library. The SD results are generated under a few flow conditions.

5. COMPUTATIONAL RESULTS

The SD results are presented in this section for both non-geometry and geometry sensitivity derivatives. The AD-generated derivatives are compared with DD-generated derivatives to verify the accurancy of the AD results. For all the results presented here, a relatively coarse mesh with 80 × 24 × 24 mesh points and a maximun number of steps of 750 are used. All AD-results are obtained under 32-bit accurancy, while DD-results are obtained under both 32-bit accurancy and 64-bit accurancy. Convergence histories of a 32-bit run and a 64-bit run are given in Figure 1 and Figure 2, respectively.

Two cases are tested for a rotor-blade in hover with tip-Mach number of 0.6288 and 0.4, respectively. The blade planform is made by Apache main rotor with wing-sections defined by

$$\begin{aligned} y_u &= gc_{i1} + gc_{i2}\sqrt{x} + gc_{i3}x + gc_{i4}x^2 + gc_{i5}x^3 + gc_{i6}x^4 \\ -y_l &= gc_{i7} + gc_{i8}\sqrt{x} + gc_{i9}x + gc_{i10}x^2 + gc_{i11}x^3 + gc_{i12}x^4 \end{aligned} \tag{6}$$

where subscripts u and l denote upper and lower surfaces, respectively; subscript i denotes blade-section number, and the coefficients gc_{i1} through gc_{i12} are twelve design variables for each blade section. The NACA0012 airfoil section is used for each blade-station, where $gc_1 = gc_7 = 0.00000$, $gc_2 = gc_8 = 0.17814$, $gc_3 = gc_9 = -0.07560$, $gc_4 = gc_{10} = -0.21096$, $gc_5 = gc_{11} = 0.17058$, and $gc_6 = gc_{12} = -0.06090$[14]. The derivatives with respect to one of these design variables, gc_{19}, are obtained along with other non-geometry derivatives as an example of geometry derivatives. Figure 3 shows blade geometry along with calculated surface pressure coefficients (C_p) for $M_{tip} = 0.6288$. Figure 4 presents calculated spanwise distribitions of lifting coefficient (C_L), drag coefficients (C_D) and moment coefficients (C_M) for the same case.

5.1 Accurancy

Table 1 and Table 2 give the AD-generated derivatives for the mentioned two testing cases. Each table has three dependent variables and three independent variables, which gives a total of nine derivatives. These results are obtained under 32-bit accurancy.

To validate the AD-generated results, a series of DD computations is made. Comparisons in terms of the ratio of AD-generated Deravatives (D_{AD}) to the DD-generated Derivatives (D_{DD}) are given in Table 3 and Table 4, where 32-bit accurancy DD runs are made; it should be noted that if AD-generated derivative is same as DD-generated derivative, this ratio is one. Relative perturbation sizes for DD runs are chosen as 10^{-1} to 10^{-6} for Table 1 and 10^{-1} to 10^{-4} for Table 2 (for derivatives with respect to α where $\alpha = 0^o$, the perturbation sizes are absolute ones as indicated in the tables). The DD-results of these two tables indicate that the DD-generated derivatives are very sensitive to the perturbation size and the accurancy of the DD-generated derivatives is hard to assess. The DD-results for small perturbation sizes are totally unacceptable, and inaccurate for large

perturbation size as compared with AD-results. The agreement of AD-generated results with DD-generated results are obtained for 2 digits (1 digit for certain derivatives) for some DD perturbation sizes.

To further validate the results, the DD runs were performed under 64-bit accurancy. Figures 1 and 2 indicate that 64-bit run converges better than that of 32-bit run with less required number of iterations. Comparisons of 32-bit AD-results with 64-bit DD-results are given in Table 5 for $M_{tip} = 0.6288$ flow case. It is seen that the DD-results with 64-bit accurancy are much less sensitive to the perturbation size compared with earlier cases of 32-bit accurancy, but the DD results are still dependent on the perturbation size and optimun perturbation sizes vary for varing derivatives. The agreement of AD-generated derivatives with the DD-generated derivatives are obtained for 3 to 4 digits for most of the perturbation sizes, although AD-results are obtained under 32-bit accurancy and DD-results are obtained under 64-bit accurancy.

5.2 Memory Requirement and Computational Efficiency

The AD code requires more memory than the original code due to additional memory required for derivatives and ADIFOR2.0 dependency anslysis. The ratio of the AD code to original code memory requirements divided by one plus number of derivatives for each dependent variable is around 1.02 in present case, assuming that the same number of bits of accurancy for each word is required.

In comparing the CPU time for AD code and original code, the ratio of the AD code to original code CPU time requirements divided by one plus number of derivatives for each dependent variable is examined. In the present case, this ratio is around 1.9. It has been seen[6] that when the number of derivatives for each dependent variable increases, this ratio decreases. It should be noted that such CUP time requirement comparison is made based on assumptions that same number of iterations is needed to get AD and DD derivatives with same number of bits of accurancy. However, it has been seen that accurate AD-derivatives can be obtained under 32-bit accuracy while accurate DD-derivatives must be obtained under 64-bit accurancy.

6. CONCLUDING REMARKS

An AD-version of the rotor flow solver FPX along with the GRGN3 grid generator is developed using ADIFOR2.0 automatic differentiation source translator. The both non-geometry and geometry DD-derivatives are computed and are compared with AD-derivatives to validate the AD version of the code. Througth this investigation, several concluding remarks can be drawn:

1. Based on the comparison of the AD-derivatives with DD-derivatives, it is believed that

the AD-version of the code produces correct and accurate SD results for both non-geometry and geometry derivatives.

2. ADIFOR2.0 is a powerful tool in translating Fortran77 code into SD code; the present AD-version of the code is generated in $O(man - month)$ as compared to $O(man - year)$ as required to generate same SD codes by other means.

3. The unified approach, where grid generator is embedded into flow slower, is an efficient way to generate accurate geometry derivatives. Agreement of AD-generated geometry derivatives with those of DD-generated are obtained for 3 to 4 digits.

4. AD-derivatives are more reliable and less expensive to obtain than DD-derivatives; an accurate AD-derivative can be obtained under 32-bit accurancy, while an accurate DD-derivative must be obtained under 64-bit accurancy with an appropriate perturbation size.

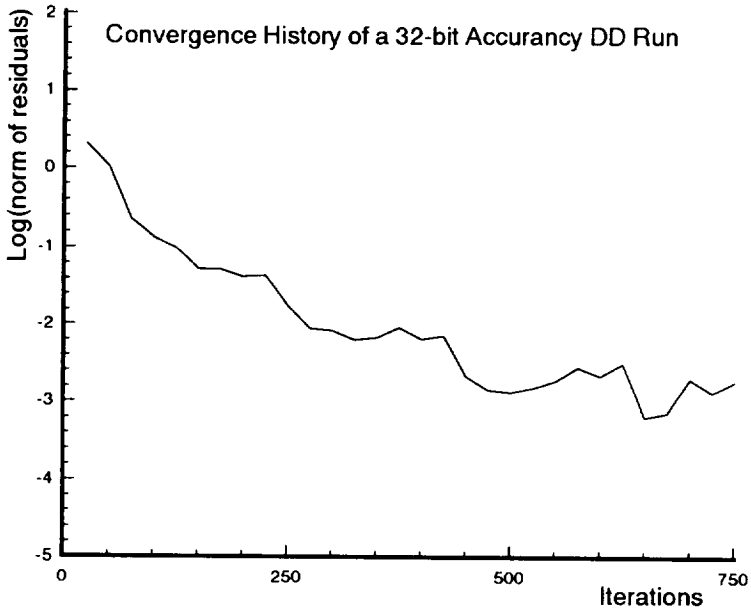

Figure 1. Convergence history of a 32-bit accurancy DD run.

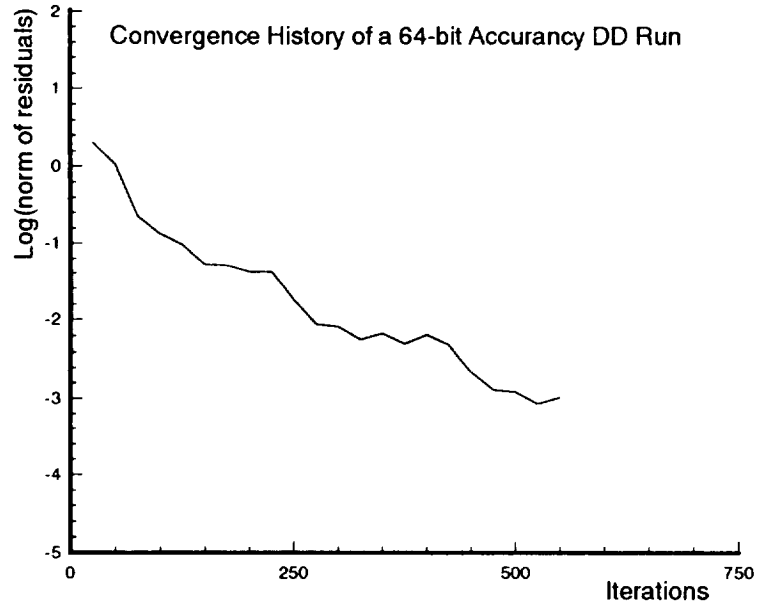

Figure 2. Convergence history of a 64-bit accurancy DD run.

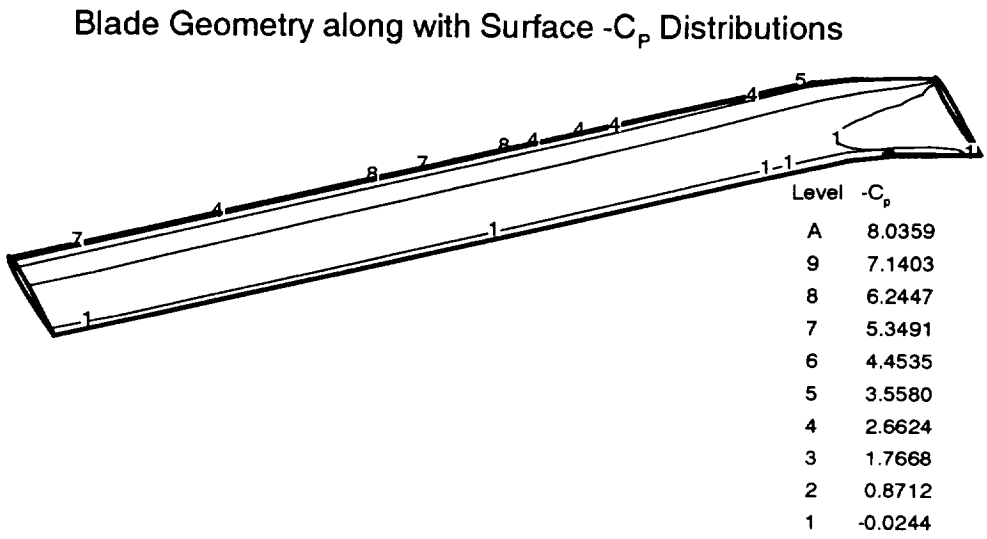

Figure 3. Blade geometry along with computed $-C_p$-distributions, $M_{tip} = 0.6288$.

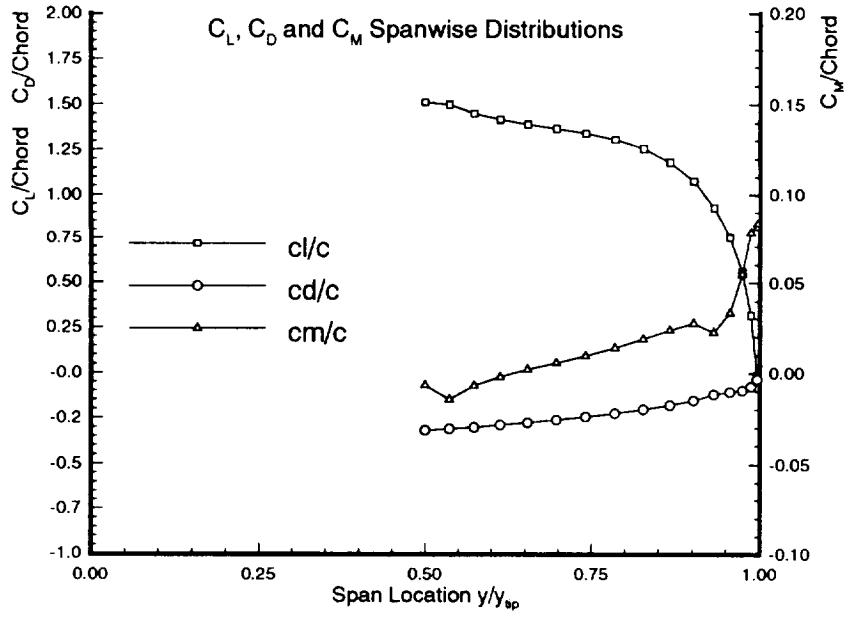

Figure 4. Computed spanwise distributions of C_L, C_D and C_M, $M_{tip} = 0.6288$.

Table 1. AD-generated Derivatives, $M_{tip} = 0.6288$.

D_{AD}	∂C_L	∂C_D	∂C_M
$\partial\alpha$	1.53826E-01	-3.74803E-02	1.07979E-03
∂M_{tip}	3.45957E-01	-3.45604E-02	4.62332E-03
∂GC_{19}	-2.07972E+01	3.90939E+00	3.67327E+00

Table 2. AD-generated Derivatives, $M_{tip} = 0.4$.

D_{AD}	∂C_L	∂C_D	∂C_M
$\partial\alpha$	1.44994E-01	-3.78407E-02	4.10811E-04
∂M_{tip}	1.72614E-01	-1.65142E-02	1.57250E-03
∂GC_{19}	-1.99362E+01	3.92898E+00	3.66895E+00

Table 3. Ratios of Sensitivity Derivatives, $M_{tip} = 0.6288$, 32-bit Accurancy for DD Runs.

$\frac{D_{AD}}{D_{DD}}$	$\frac{\partial C_L}{\partial \alpha}$ Ratio	$\frac{\partial C_D}{\partial \alpha}$ Ratio	$\frac{\partial C_M}{\partial \alpha}$ Ratio
$abs(\Delta) = 10^{-1}$	0.99895	0.99893	0.98858
$abs(\Delta) = 10^{-2}$	0.99883	1.00026	1.01770
$abs(\Delta) = 10^{-3}$	0.96586	0.96223	-0.18267
$abs(\Delta) = 10^{-4}$	0.72493	0.71865	-0.01487
$abs(\Delta) = 10^{-5}$	0.19259	0.23289	-0.00198
$abs(\Delta) = 10^{-6}$	0.01870	0.03593	-0.00019
$\frac{D_{AD}}{D_{DD}}$	$\frac{\partial C_L}{\partial M_{tip}}$ Ratio	$\frac{\partial C_D}{\partial M_{tip}}$ Ratio	$\frac{\partial C_M}{\partial M_{tip}}$ Ratio
$\Delta = 10^{-1}$	0.90151	0.88662	0.85380
$\Delta = 10^{-2}$	0.98864	0.98753	1.01533
$\Delta = 10^{-3}$	0.99068	1.00163	1.23405
$\Delta = 10^{-4}$	0.72128	0.62324	-0.03855
$\Delta = 10^{-5}$	0.26069	0.16573	-0.00646
$\Delta = 10^{-6}$	0.06085	0.04863	-0.00149
$\frac{D_{AD}}{D_{DD}}$	$\frac{\partial C_L}{\partial GC_{19}}$ Ratio	$\frac{\partial C_D}{\partial GC_{19}}$ Ratio	$\frac{\partial C_M}{\partial GC_{19}}$ Ratio
$\Delta = 10^{-1}$	0.99036	0.99900	1.01637
$\Delta = 10^{-2}$	0.99884	0.99995	0.99991
$\Delta = 10^{-3}$	0.99699	0.99498	0.97617
$\Delta = 10^{-4}$	0.95782	0.95724	0.87752
$\Delta = 10^{-5}$	0.83917	0.87452	0.64505
$\Delta = 10^{-6}$	0.14848	0.13118	0.04001

Table 4. Ratios of Sensitivity Derivatives, $M_{tip} = 0.4$, 32-bit Accurancy for DD Runs.

$\frac{D_{AD}}{D_{DD}}$	$\frac{\partial C_L}{\partial \alpha}$ Ratio	$\frac{\partial C_D}{\partial \alpha}$ Ratio	$\frac{\partial C_M}{\partial \alpha}$ Ratio
$abs(\Delta) = 10^{-1}$	0.99934	0.99806	0.98837
$abs(\Delta) = 10^{-2}$	0.99827	0.99907	1.07064
$abs(\Delta) = 10^{-3}$	0.98805	0.99430	-1.82275
$abs(\Delta) = 10^{-4}$	0.83308	0.63486	-0.04225
$\frac{D_{AD}}{D_{DD}}$	$\frac{\partial C_L}{\partial M_{tip}}$ Ratio	$\frac{\partial C_D}{\partial M_{tip}}$ Ratio	$\frac{\partial C_M}{\partial M_{tip}}$ Ratio
$\Delta = 10^{-1}$	0.93844	0.93752	0.92640
$\Delta = 10^{-2}$	0.99296	0.98599	1.01363
$\Delta = 10^{-3}$	0.97181	0.96370	-5.89857
$\Delta = 10^{-4}$	0.83942	0.79161	-0.09433
$\frac{D_{AD}}{D_{DD}}$	$\frac{\partial C_L}{\partial GC_{19}}$ Ratio	$\frac{\partial C_D}{\partial GC_{19}}$ Ratio	$\frac{\partial C_M}{\partial GC_{19}}$ Ratio
$\Delta = 10^{-1}$	0.99122	0.99291	1.01455
$\Delta = 10^{-2}$	0.99928	0.99935	1.00124
$\Delta = 10^{-3}$	0.99898	0.99717	0.99709
$\Delta = 10^{-4}$	0.96734	0.91187	0.94093

Table 5. Ratios of Sensitivity Derivatives, $M_{tip} = 0.6288$, 64-bit Accurancy for DD Runs.

$\frac{D_{AD}}{D_{DD}}$	$\frac{\partial C_L}{\partial \alpha}$ Ratio	$\frac{\partial C_D}{\partial \alpha}$ Ratio	$\frac{\partial C_M}{\partial \alpha}$ Ratio
$abs(\Delta) = 10^{-1}$	0.99908	0.99864	0.98876
$abs(\Delta) = 10^{-2}$	0.99988	0.99958	1.00305
$abs(\Delta) = 10^{-3}$	0.99995	0.99968	1.00449
$abs(\Delta) = 10^{-4}$	0.99996	0.99969	1.00464
$abs(\Delta) = 10^{-5}$	0.99996	0.99969	1.00464
$abs(\Delta) = 10^{-6}$	0.99996	0.99969	1.00446
$abs(\Delta) = 10^{-7}$	0.99997	0.99974	0.99981
$abs(\Delta) = 10^{-8}$	0.99952	1.00215	0.98163
$\frac{D_{AD}}{D_{DD}}$	$\frac{\partial C_L}{\partial M_{tip}}$ Ratio	$\frac{\partial C_D}{\partial M_{tip}}$ Ratio	$\frac{\partial C_M}{\partial M_{tip}}$ Ratio
$\Delta = 10^{-1}$	0.90222	0.89251	0.85847
$\Delta = 10^{-2}$	0.99286	1.00082	0.85241
$\Delta = 10^{-3}$	0.99938	1.00437	1.00711
$\Delta = 10^{-4}$	1.00028	1.00544	1.00851
$\Delta = 10^{-5}$	1.00037	1.00555	1.00873
$\Delta = 10^{-6}$	1.00038	1.00558	1.00873
$\Delta = 10^{-7}$	1.00036	1.00595	1.00975
$\Delta = 10^{-8}$	1.00024	1.00609	1.00438
$\frac{D_{AD}}{D_{DD}}$	$\frac{\partial C_L}{\partial GC_{19}}$ Ratio	$\frac{\partial C_D}{\partial GC_{19}}$ Ratio	$\frac{\partial C_M}{\partial GC_{19}}$ Ratio
$\Delta = 10^{-1}$	0.99021	0.99862	1.01581
$\Delta = 10^{-2}$	0.99874	0.99960	1.00008
$\Delta = 10^{-3}$	0.99971	0.99989	0.99877
$\Delta = 10^{-4}$	0.99981	0.99992	0.99864
$\Delta = 10^{-5}$	0.99982	0.99992	0.99863
$\Delta = 10^{-6}$	0.99982	0.99993	0.99863
$\Delta = 10^{-7}$	0.99982	0.99995	0.99863

REFERENCES

1. Bischof, C., Corliss, G., Green, L., Griewank, A., Haigler, K., and Newman, P., "Automatic Differentiation of Advanced CFD Codes for Multidisciplinary Design", *Computing Systems in Engineering*, Vol. 3, No. 6, 1992, pp. 625-637.

2. Green, L., Newman, P., and Haigler, K., "Sensitivity Derivatives for Advanced CFD Algorithm and Viscous Modelling Parameters via Automatic Differentiation", AIAA Paper 93-3321, 1993.

3. Green, L., Carle, A., Bischof, C., Haigler, K., and Newman, P.,"Sensitivity Derivatives For Multidisciplinary Design Optimization via Automatic Differentiation, II", lecture notes on ADIFOR Workshop, NASA Langley Research Center, Hampton, Virginia, 1993.

4. Korivi, V., Taylor, A., Newman, P., Hou, G. and Jones, H., "An Approximately Factored Incremental Strategy for Calculating Consistent Discrete Aerodynamic Sensitivity Derivatives", *Journal of Computational Physics*, Vol. 113, No. 2, 1994, pp.336-346.

5. Hou, G., Maroju, V., Taylor, A., Korivi, V., and Newman, P., "Transonic Turbulent Airfoil Design Optimization with Automatic Differentiation in Incremental Iterative Forms", AIAA Paper 95-1692, 1995.

6. Hu, H., "Sensitivity Derivatives of the FLOMGEM CFD Code via Automatic Differentiator ADIFOR2.0 ", Technical Report, DAAJ02-94-C-0032, U.S. Army Aviation Applied Technology Directorate, Ft. Eustis, Virginia, July 1996.

7. Bridgeman, J. O., Prichard, D. and Caradonna, F. X., "The Development of A CFD Potential Method for The Analysis of Tilt-Rotors", the Proceedings of the AHS Technical Specialists Meeting on Rotorcraft Acoustics and Fluid Dynamics, Philadelphia, PA, October 15-17, 1991.

8. Strawn, R. and Caradonna, F. X., "Conservative Full-Potential Model for Unsteady Transonic Rotor Flows", AIAA Journal, Vol. 25, No. 2, 1987, pp. 193-198.

9. Caradonna, F. X., Strawn, R. C. and Bridgeman, J. O., "An Experimental and Computational Study of Rotor-Vortex Interactions", the Proceedings of the Fourteenth European Rotorcraft Forum, Milano, Italy, September 20-23, 1988.

10. Purcell, T., "A Prediction of High-Speed Rotor Noise", AIAA Paper -89-1130, presented at the AIAA 12th Aeroacoustic Conference, San Antonio, Texas, April 10-12, 1989.

11. Strawn, R. C. and Bridgeman, J. O., "An Improved Three-Dimensional Aerodynamics

Model for Helicopter Airloads Prediction", the AIAA 29th Aerospace Sciences Meeting, Reno, Nevada, January 7-10, 1991.

12. Bridgeman, J. O., Ramachandran, K., Caradonna, F. X. and Prichard, D., "A Computational Analysis of Parallel Blade-Vortex Interactions Using Vorticity Embedding", the Proceedings of the 50th AHS Annual Forum, Washington, DC, May 11-13, 1994, pp. 1197-1210.

13. Bischof, C., Carle, A., Khademi, P., and Mauer, A., "The ADIFOR2.0 System for the Automatic Differentiation of Fortran 77 Programs", Argonne Preprint ANL-MCS-P481-1194; also as CRPC Technical Report CRPC-TR 94491.

14. Abbott, I., and von Doenhoff, A., Theory of Wing Sections, Dover Publications, New York, 1959.